CLAIMS

What is claimed is:

1. A method of managing resources in a multithreaded processor, the method comprising:

partitioning a resource into a number of portions based upon a number of threads being executed concurrently; and

performing resource allocation for each thread in its respective portion of the resource.

2. The method of claim 1 wherein partitioning comprises:

sizing the corresponding portion for each thread according to a partitioning scheme; and

marking the corresponding portion as being reserved for the respective thread.

3. The method of claim 2 wherein the size of each portion is determined based upon at least one factor selected from the group consisting of a first factor indicating the number of threads being executed concurrently, a second factor indicating the capacity of the resource, and a third factor indicating a relative processing priority of each thread.

4. The method of claim 2 wherein marking comprises:

specifying the lower and upper boundaries of each portion corresponding to its respective location within the resource.

5. The method of claim 1 further comprising:

initializing each portion of the resource in response to one or more signals indicating a mode transition.

6. The method of claim 5 wherein the mode transition is invoked in response to an event or condition.

7. The method of claim 5 wherein initializing comprises:

initializing a set of pointers corresponding to the respective portion.

8. The method of claim 7 wherein the set of pointers comprises a first pointer used to keep track of entries that have been allocated in the respective portion and a second pointer used to keep track of entries that have been deallocated in the respective portion.

9. The method of claim 1 wherein performing resource allocation for each thread comprises:

performing stall computation for each thread to determine whether the respective portion has sufficient available entries to allocate a number of entries required for the execution of one or more instructions from the respective thread; and

allocating the number of entries required in the respective portion if the respective portion has sufficient available entries.

10. The method of claim 9 wherein performing stall computation for each thread is done in parallel with performing stall computation for another thread.

11. The method of claim 9 wherein performing stall computation for each thread and performing stall computation for another thread are multiplexed.

12. The method of claim 9 wherein allocating the number of entries required for each thread is done in parallel with allocating a number of entries required for another thread.

13. The method of claim 9 wherein allocating the number of entries required for each thread and allocating a number of entries required for another thread are multiplexed.

14. The method of claim 9 wherein performing stall computation for each thread comprises:

determining the number of entries to be allocated for the one or more instructions from the respective thread;

determining a number of entries available in the respective portion; and

comparing the number of entries to be allocated with the number of entries available in the respective portion.

15. The method of claim 14 further comprising:

activating one or more stall signals if the number of entries required exceeds the number of entries available in the respective portion, the one or more stall signals indicating that the one or more instructions from the respective thread cannot be executed due to insufficient available resource in the respective portion.

16. The method of claim 14 wherein determining the number of entries to be allocated for the one or more instructions comprises:

determining the type of the one or more instructions; and

determining whether the resource is needed to execute the one or more instructions based upon the type of the one or more instructions.

17. The method of claim 16 wherein the number of entries to be allocated is greater than the number of entries needed to execute the one or more instructions.

18. The method of claim 14 wherein determining the number of entries available comprises:

comparing the value of the first pointer with the value of the second pointer to determine the number of entries that are available for allocation.

19. The method of claim 18 further comprising:

wrapping the first pointer when it is advanced past the end of the respective portion.

20. The method of claim 19 including:

updating a wrap bit to indicate that the first pointer is wrapped around.

21. The method of claim 18 further comprising:

wrapping the second pointer when it is advanced past the end of respective portion.

22. The method of claim 21 including:

updating a wrap bit to indicate that the second pointer is wrapped around.

23. A method of managing a resource in a multithreaded processor, the method comprising:

detecting a signal indicating a processing mode;

performing resource allocation according to a multithread scheme if the processing mode is multithreading; and

performing resource allocation according to a single thread scheme if the processing mode is single threading.

24. The method of claim 23 wherein the signal indicating the processing mode is updated in response to an occurrence of an event or a condition.

25. The method of claim 23 wherein performing resource allocation according to the single thread scheme comprises:

determining which thread is active; and

assigning all of the resource to the thread which is active.

26. The method of claim 25 including:

performing resource allocation for the thread which is active.

27. The method of claim 26 wherein performing resource allocation for the active thread comprises:

initializing pointers of the active thread in response to one or more signals indicating a mode transition;

receiving a set of instructions from the active thread;

determining whether the resource has sufficient available entries to allocate for the set of instructions from the active thread; and

allocating an amount of entries needed for the set of instructions if the resource has sufficient available entries to allocate.

28. The method of claim 27 further comprising:

activating one or more stall signals for the active thread if the resource does not have sufficient available entries to allocate, the one or more stall signals indicating that the set of instructions cannot be executed due to insufficient resource.

29. The method of claim 27 wherein determining whether the resource has sufficient available entries comprises:

computing the amount of entries to be allocated for the set of instructions;

computing the amount of available entries in the resource; and

comparing the amount to be allocated with the amount of available entries.

30. The method of claim 23 wherein performing resource allocation according to the multithread scheme comprises:

partitioning the resource into a number of portions corresponding to a number of threads being executed concurrently; and

performing resource allocation for each thread in its respective portion of the resource.

31. The method of claim 30 wherein the size of each portion is predetermined.

32. The method of claim 30 wherein the size of each portion is determined based upon at least one factor selected from the group consisting of a first factor indicating a number of active threads executed concurrently, a second factor indicating the capacity of the resource, and a third factor indicating a relative processing priority assigned to each thread.

33. The method of claim 30 wherein performing resource allocation comprises:

initializing pointers corresponding to each portion in response to one or more signals indicating a mode transition;

receiving a set of instructions from the respective thread; and

determining whether the respective portion has sufficient available entries to allocate for the set of instructions from the respective thread.

34. The method of claim 33 including:

activating one or more stall signals for the respective thread if the respective portion does not have sufficient available entries to allocate.

35. The method of claim 34 further comprising:

determining whether the set of instructions belongs to the respective thread; and

allocating an amount of entries needed for the set of instructions in the respective portion if the set of instructions belongs to the respective thread and the one or more stall signal for the respective thread is not activated.

36. The method of claim 35 wherein determining whether the set of instructions belongs to the respective thread comprises:

examining the value of a thread bit associated with the respective instruction, the value of the thread bit indicating which thread the respective instruction belongs.

37. The method of claim 33 wherein determining whether the respective portion has sufficient available entries comprises:

computing an amount of entries to be allocated for the set of instructions;

computing an amount of available entries in the respective portion; and

comparing the amount of entries to be allocated with the amount of available entries.

38. The method of claim 34 including:

stalling further fetching of instructions from the respective thread if the one or more stall signals for the respective thread is activated.

39. An apparatus for managing a resource in a multithreaded processor, the apparatus comprising:

partition logic to partition the resource into a number of portions corresponding to a number of threads being executed concurrently; and

resource control logic to perform resource allocation for each thread in its respective portion of the resource.

40. The apparatus of claim 39 wherein the size of each portion is predetermined.

41. The apparatus of claim 39 wherein the size of each portion is determined based upon at least one factor selected from the group consisting of a first factor indicating the number of threads being executed concurrently, a second factor indicating the capacity of the resource, and a third factor indicating a relative processing priority of each thread.

42. The apparatus of claim 39 further comprising:

initialization logic to initialize pointers corresponding to each respective portion in response to one or more signals indicating a mode transition.

43. The apparatus of claim 42 wherein the initialization logic updates the pointers to point to the respective portion of the resource.

44. The apparatus of claim 43 wherein the pointers comprise a first pointer used to keep track of entries that have been allocated in the respective portion and a second pointer used to keep track of entries that have been deallocated in the respective portion.

45. The apparatus of claim 39 wherein the resource control logic comprises:

stall computation logic to determine whether the respective portion has sufficient available entries to allocate an amount of entries for a set of instructions from the first thread; and

allocation logic to allocate the amount of entries for the set of instructions in the respective portion if the respective portion has sufficient available entries to allocate.

46. The apparatus of claim 39 wherein stall computations for the respective threads are performed in parallel and resource allocation for the respective threads are performed in parallel.

47. The apparatus of claim 39 wherein stall computations for the respective threads are performed in parallel and resource allocation for the respective threads are performed in a multiplexed manner.

48. The apparatus of claim 45 wherein the stall computation logic comprises:

resource requirement logic to compute an amount of entries to be allocated for the set of instructions from the respective thread;

resource availability logic to determine an amount of entries available in the respective portion; and

comparison logic to compare the amount of entries to be allocated with the amount of entries available.

49. The apparatus of claim 48 further comprising:

stall activation logic to activate one or more stall signals if the amount to be allocated exceeds the amount of entries available, the one or more stall signals indicating that the set of instruction from the respective thread cannot be executed due to insufficient available resource in the respective portion.

50. The apparatus of claim 48 wherein the resource availability logic comprises:

first tracking logic to keep track of the amount of entries in the respective portion that have been allocated; and

second tracking logic to keep track of the amount of entries in the respective portion that have been deallocated.

51. An apparatus for controlling usage of a resource in a multithreaded processor, the apparatus comprising:

detection logic to detect a signal indicating a processing mode; and

a control circuit to perform resource allocation according to a single thread scheme if the processing mode is single threading and to perform resource allocation according to a multithread scheme if the processing mode is multithreading.

52. The apparatus of claim 51 wherein the control circuit comprises resource partition logic to partition the resource into a number of portions according to a number of threads being executed concurrently.

53. The apparatus of claim 52 wherein the resource partition logic assigns all of the resource to the thread that is active if the processing mode is single threading.

54. The apparatus of claims 52 wherein the resource partition logic assigns a portion of the resource to each of the threads being executed concurrently if the processing mode is multithreading.

55. The apparatus of claim 52 wherein the control circuit further comprises:

resource allocation logic to perform resource allocation based upon the number of threads being executed concurrently.

56. The apparatus of claim 55 wherein the control circuit comprises:

initialization logic to initialize the resource based upon the number of threads being executed concurrently.

57. The apparatus of claim 55 wherein the resource allocation logic comprises:

resource requirement logic to compute an amount of entries to be allocated for a set of instructions;

resource availability logic to compute an amount of available entries in the resource; and

allocate logic to allocate the amount of entries for the set of instructions in the resource if the amount of available entries is sufficient.

58. The apparatus of claim 57 wherein the resource availability logic computes the amount of available entries with respect to the entire resource if the processing mode is single threading.

59. The apparatus of claim 57 wherein the resource availability logic computes the amount of available entries with respect to the portion assigned to the respective thread if the processing mode is multithreading.

60. The apparatus of claim 57 wherein the allocate logic comprises logic to allocate the amount of entries in the corresponding portion if the set of instructions belongs to the respective thread and the corresponding portion has enough available entries.

61. The apparatus of claim 57 including:

stall activation logic to activate at least one stall signal if the amount of entries available is not sufficient.

62. A processor comprising:

an instruction delivery engine to store and fetch instructions either from one or more threads based upon a current processing mode; and

an allocator to receive instructions from the instruction delivery engine and to perform allocation in a resource based upon the current processing mode.

63. The processor of claim 62 wherein the allocator assigns the entire resource to the thread that is active if the current processing mode is single threading.

64. The processor of claim 62 wherein the allocator assigns a portion of the resource to each of the threads running concurrently if the current processing mode is multithreading.

65. The processor of claim 63 wherein the allocator allocates an amount of entries for the instructions from the active thread in the resource if the resource has sufficient available entries and wherein the allocator activates at least one stall signal if the resource does not have sufficient available entries.

66. The processor of claim 64 wherein the allocator allocates an amount of entries for the instructions from each respective thread in the respective portion if the respective portion has sufficient available entries and wherein the allocator activates at least one stall signal if the respective portion does not have sufficient available entries.

67. The processor of claim 66 wherein the instruction delivery engine uses the at least one stall signal to perform its corresponding function.

68. The processor of claim 67 wherein the instruction delivery engine re-fetches the stalled instructions in the respective thread to the allocator if the at least one stall signal is activated.

69. The processor of claim 67 wherein the instruction delivery engine fetches a subsequent instruction from another thread to the allocator if the at least one stall signal for the respective thread is activated and said another thread is not stalled.

70. The processor of claim 67 wherein the instruction delivery engine fetches an invalid instruction to the allocator if the stall signal for the respective thread is activated.

71. An apparatus for managing a resource in a multithreaded processor, the apparatus comprising:

means for assigning a portion of the resource to each of a plurality of threads being executed concurrently in the multithreaded processor; and

means for performing resource allocation for each respective thread in its respective portion of the resource.

72. The apparatus of claim 71 further comprising:

means for initializing each respective portion in response to a signal indicating a mode transition.

73. The apparatus of claim 72 wherein means for initializing comprises:

means for setting a corresponding set of pointers to point to the respective portion.

74. The apparatus of claim 71 wherein means for performing resource allocation for the respective thread comprises:

means for performing stall computation for the respective thread to determine whether the respective portion has sufficient available entries to allocate an amount of entries for the execution of a set of instructions from the respective thread; and

means for allocating the amount of entries in the respective portion for the set of instructions if the respective portion has sufficient available entries to allocate.

75. The apparatus of claim 74 wherein means for performing stall computation comprises:

means for determining the amount of entries to be allocated for the set of instructions;

means for determining an amount of entries available in the respective portion; and

means for comparing the amount of entries to be allocated with the amount of entries available.

76. The apparatus of claim 75 further comprising:

means for activating at least one stall signal if the amount to be allocated exceeds the amount of entries available, the at least one stall signal indicating that the set of instructions from the respective thread cannot be executed due to insufficient available resource.

77. The apparatus of claim 75 wherein means for determining the amount of entries available comprises:

means for keeping track of the amount of entries in the respective portion that have been allocated; and

means for keeping track of the amount of entries in the respective portion that have been deallocated.

78. An apparatus for controlling usage of a resource, the apparatus comprising:

detection means for detecting a signal indicating a processing mode; and

control means for performing resource allocation according to a single thread scheme if the processing mode is single threading and for performing resource allocation according to a multithread scheme if the processing mode is multithreading.

79. The apparatus of claim 78 wherein the control means comprises:

partition means for partitioning the resource into a number of portions based upon a number of threads being executed concurrently.

80. The apparatus of claim 79 wherein the control means further comprises:

allocation means for allocating the resource based upon the number of threads being executed concurrently.

81. The apparatus of claim 80 wherein the control means comprises:

initialization means for initializing the resource based upon the number of threads being executed concurrently.

82. The apparatus of claim 80 wherein the allocation means comprises:

requirement computing means for computing an amount of entries to be allocated for a set of instructions;

availability computing means for computing an amount of available entries in the resource; and

means for allocating the amount of entries in the resource if the amount of available entries is sufficient.

83. The apparatus of claim 82 including:

stall activation means for activating at least one stall signal if the amount of entries available is not sufficient.